CTCSS Nachrüstung für ein Yaesu 221

Mein bei Ebay erstandenes 2m-Funkgerät hatte kein CTCSS. Ich muss für den Relais-Betrieb eine Schaltung nachrüsten. Und kommt hinzu, dass die genaue Einstellung der Frequenzen allein mit der Analogskala ziemlich frickelig ist. Daran kann ich mich nur schwer gewöhnen. Also muss auch eine digitale Frequenzanzeige her.

CTCSS

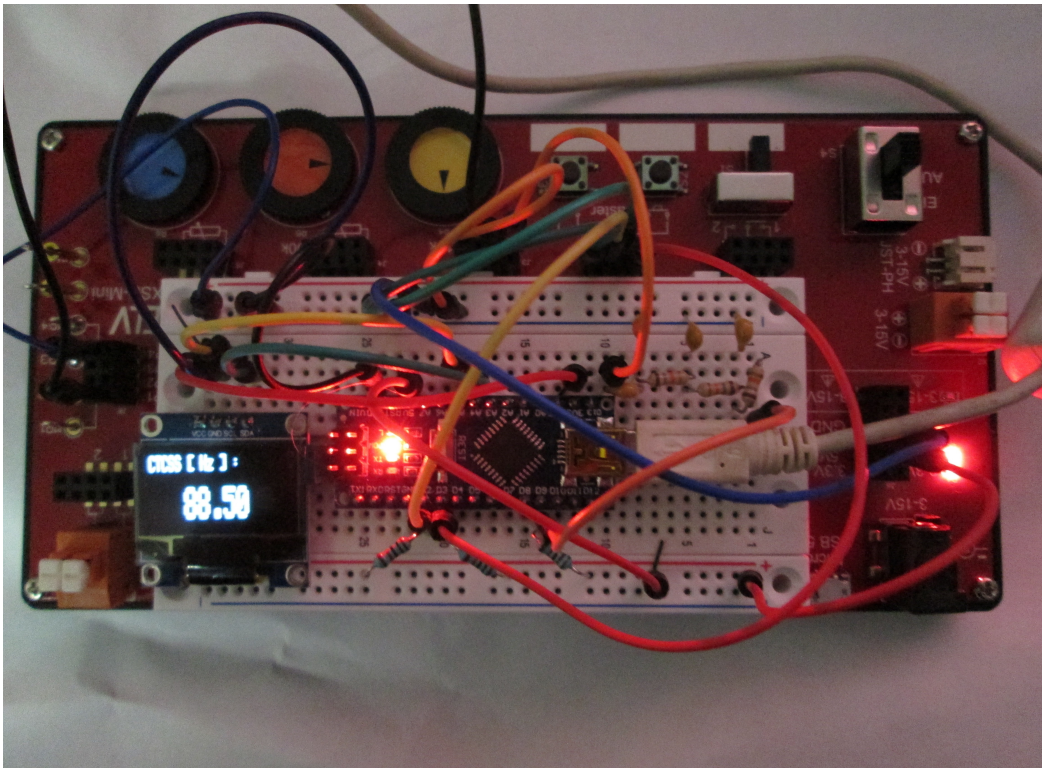
Ich stand vor der Wahl: Entweder kaufen oder selbst bauen. Die Angebote waren mir zu teuer. Und an mehreren Schaltern die Frequenz einzustellen, war mir auch zu umständlich. Da lagen ja noch Arduino-Nanos in der Bastelkiste. Und bald entdeckte ich den Bauvorschlag von ON7GF. Er sieht allerdings eine Anzeige nur über den seriell angesteuerten Monitor auf dem PC vor. Ich wollte aber am Gerät sehen, was ist. Und so suchte ich das letzte vorhandene Oled-Display. Es war ein Typ 1306. Mit 0,96 Zoll groß genug, um einige Ziffern darzustellen.

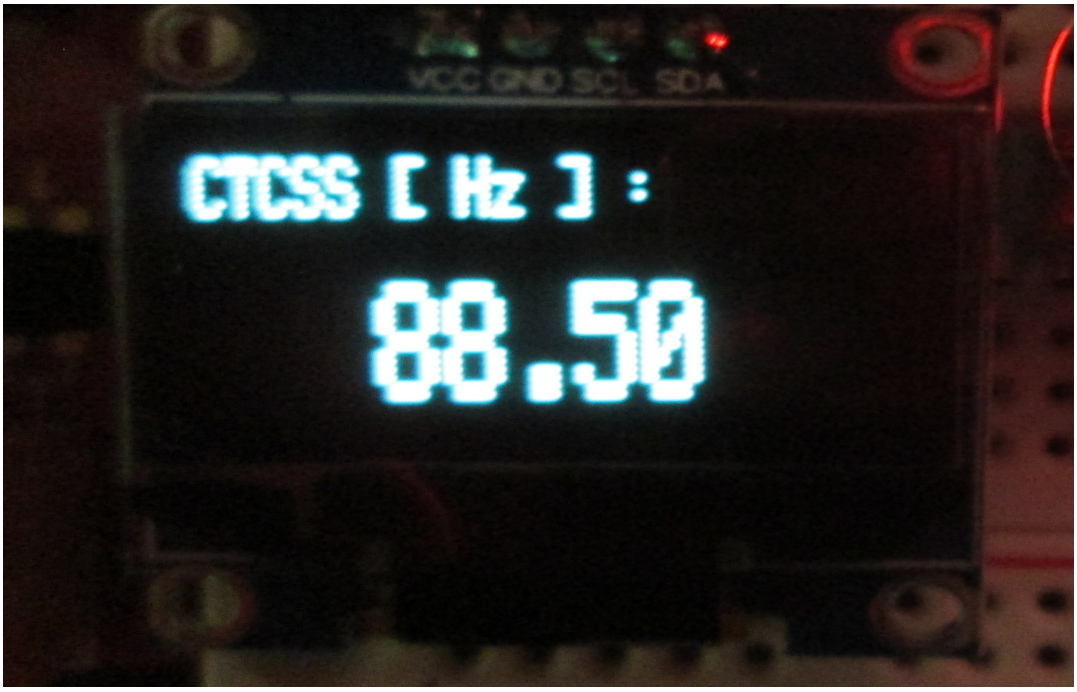
Zunächst musste ich mich mal wieder mit dem Interruptbetrieb am Arduino vertraut machen. Denn wenn man da die falschen Pins anschließt, tut sich nichts. Nur D2 und D3 reagieren. Na schön, ich programmierte mal eben die LED13 für die Kontrolle der richtigen Funktion. Denn am Display wurde immer wieder derselbe Text unverändert angezeigt. Man sucht im Programm nach logischen Fehlern. Doch vergebens, wenn man nicht weiß, dass jede Ausgabe von Texten mit dem Befehl display.display(); enden muss. Das lernte ich dann, nachdem ich mich im Netz schlau machte. Nach einiger Programmierzeit lief dann alles wie gewünscht. Vielleicht nicht ganz: Wenn man häufiger die Frequenz wechseln muss, empfehle ich den Einsatz eines Impulsgebers. Denn man kann hier bis zu 50 Frequenzen anwählen. Das geht zwar mit den beiden Tastern UP / DOWN ganz leicht, doch bei größerer Distanz zum Anfangswert werden

die Finger wund.

Startanzeige
Beim Start sollte eigentlich die zuletzt gewählte Frequenz wieder automatisch erscheinen. Das erfordert allerdings Speicherbetrieb. Die Sache wird dann aber kompliziert. Vielleicht findet sich jemand für eine Änderung der Software. Ich habe vorgesehen, dass man per PC den Arduino so flasht, dass man die gewünschte Anfangsfrequenz nach dem Einschalten erhält. Für meinen stationären Betrieb ist das die CTCSS für das Relais Böllstein auf 2m: 88,5 Hz.

Output
Der Pin D9 gibt ein Rechteck heraus. Man muss ein Filter nachschalten, damit man annähernd Sinusform erhält. Da ich plaane, diesen CTCSS-Generator zusammen mit einer digitalen Frequenzanzeige in ein eigenen Gehäuse zu bauen, kann ich im Augenblick nur das Entwicklungsmuster vorstellen. Es wurde auf dem Experimentierbrett vom ELV-Verlag zusammengestellt.





Das primitive Filter ist ein Tiefpass mit 3 x 18k und 3 x 100n bzw 1 x 1u. Hier kann man auch Induktivitäten einsetzen. Die Ausgangsspannung ist 200mV RMS. Also genügend für eine Modulation. Zur Anpassung werde ich ein Poti spendieren. Und so kommen doch einige Bauelemente zusammen, wenn man die Einheit in den Mikrofonkreis einschleifen will. Buchsen und Stecker und auch einen Anschluss für die Stromversorgung(+5V).

Software

ON7GF erlaubt die Verwendung seiner Software für private Zwecke. Und so erlaube ich ebenfalls die Erweiterung von mir. Ist ja keine große Leistung, sich die Programmteile zusammen zu suchen. Und ganz bequeme OMs bringen vielleicht noch weitere Features dazu.

Hier nun meine erste Version:

```
**********************************************************************
**************************
 *                                                          *
```

```
 *                ATMEGA328P HAM RADIO CTCSS ENCODER
*
 *                      Version: V2.0                    *
 *                                                       *
 *           Created by Geert Vanhulle - ON7GF - November 2018
*
 *                                                       *
 * Released under GPL3 - GNU GENERAL PUBLIC LICENSE - Version 3, 29 June 2007   *
 *                                                       *
 *    Derivative work can only be distributed under these same license terms    *
 *    For the full license text goto: fsf.org - The Free Software Foundation    *
 *                                                       *

**********************************************************************
**************************/


/* ---------------------------------------------------------------------
 *
 *  dit is nog niet de uiteindelijke versie. De output van deze versie is een
 *  blokgolf signaal op pin D9 met de juiste frequentie, doch een blokgolf is
 *  niet 100% bruikbaar als ctcss omdat de harmonischen in het signaal
 *  een ctcss-decoder in de war kunnen brengen.
 *
 *  De finale versie moet dus direct een sinus kunnen produceren.
 *  An D9 ist der Output des Rechtecksignals
 *  Mit D2 und D3 UP DOWN wird die Frequenz eingestellt
 *
 *   Diese Ergänzung zeigt die Frequenz des Subtones auf einem
```

```
Oled1306 an
 *   Das Programm  darf mit meiner Zustimmung privat genutzt werden
 *   DF8ZR; August 2021
 *
 *   Man könnte für das Auswählen auch einen Impulsgeber einsetzen. Macht aber nur
 *   Sinn, wenn man ein älteres Fubkgerät atändig an wechselnden Orten mobil betreibt.
   ----------------------------------------------------------------------
*/
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define OLED_RESET 4 // not used / nicht genutzt bei diesem Display

#define DRAW_DELAY 118
#define D_NUM 47

Adafruit_SSD1306 display(OLED_RESET);


int i;




// tabel gegevens
byte ocrL[50]; byte ocrH[50];

// ctcss tonen
float ctcssFreq[] =   {  67.0,    69.3,    71.9,    74.4,    77.0,   79.7,
```

```
82.5,   85.4,   88.5,   91.5,
                 94.8,   97.4,   100.0,   103.5,   107.2,  110.9,
114.8,   118.8,   123.0,   127.3,
                131.8,   136.5,   141.3,   146.2,   151.4,  156.7,
159.8,   162.2,   165.5,   167.9,
                171.3,   173.8,   177.3,   179.9,   183.5,  186.2,
189.9,   192.8,   196.6,   199.5,
                203.5,   206.5,   210.7,   218.1,   225.7,  229.1,
233.6,   241.8,   250.3,    254.1
};

// isr vlaggen
bool up = false;
bool dn = false;

// actieve toon uit de lijst
int toon=0;


// interrupt vector up
ISR(INT0_vect){
  up=true;
}

// interrupt vector dn
ISR(INT1_vect){
  dn=true;
}


void setup() {

 ctcsscalc();

 /* frequentie-deler */
```

```
// configuratie Timer1
TCCR1A = 0x50;
TCCR1B = 0x0A;
TCCR1C = 0x00;

// zet pin PB1 = OCR1A = "D9" als output
DDRB |= 1 << PB1;

// zet het deeltal in het output compare register
OCR1AH = ocrH[0];
OCR1AL = ocrL[0];

/* interrupts */
// pin 2 & 3 als input
DDRD &= ~(1 << DDD2);
DDRD &= ~(1 << DDD3);

// falling edge
EICRA |=  (1 << ISC11);
EICRA &= ~(1 << ISC10);
// falling edge
EICRA |=  (1 << ISC01);
EICRA &= ~(1 << ISC00);

// enable interrupt
EIMSK |= (1 << INT0);
EIMSK |= (1 << INT1);

/* led's als display welke toon */
// PC0 (A4) ... PC5 (A0) = uitgang
DDRC  = B00111111;
PORTC = B00000000;

Serial.begin(9600);
```

```
  // initialize with the I2C addr 0x3C / mit I2C-Adresse 0x3c
initialisieren
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);


//Wunschfrequenz beim Einschalten setzen; hier z.B.: 88,5 Hz
Böllstein-Relais 2m

   toon=8;

    OCR1AH = ocrH[toon];
    OCR1AL = ocrL[toon];

    //up=false;
    //dn=false;

  anzeige();


}



// bereken deeltal ocrnx voor elke frequentie
void ctcsscalc() {

  int N = 8;
  int clk = 16; // 16 MHz clock

  for (int n=0; n<50; n++) {

    int ocrLH = ( clk * 1000000 ) / (( 2 * N * ctcssFreq[n] ) - 1 );
```

```
    ocrH[n] = ocrLH >> 8;
    ocrL[n] = ocrLH & 0x00FF;

  }
}


 void anzeige()
  {

display.clearDisplay();

display.setTextColor(WHITE);
display.setTextSize(1);
  // set text cursor position / Textstartposition einstellen
display.setCursor(1,0);
  // show text / Text anzeigen
display.println("CTCSS [ Hz ] : ");

display.setTextSize(2);
display.setCursor(34,15);


    String Frequenz = "";     // empty string
    Frequenz.concat((ctcssFreq[toon]));



display.println(Frequenz);

display.display();//damit wird der Textaufabau auch tatsächlich
erst zum oled gesendet!


}
```

```
void loop() {



  if (up) {
    for (double wait=0; wait<25000; wait++) { int a=0; } //
debounce
    toon++;
    if (toon>49) {toon=0;}
    OCR1AH = ocrH[toon];
    OCR1AL = ocrL[toon];

    up=false;

anzeige();
  }
  if (dn) {
    for (double wait=0; wait<25000; wait++) { int a=0; } //
debounce
    toon--;
    if (toon<0) {toon=49;}
    OCR1AH = ocrH[toon];
    OCR1AL = ocrL[toon];

    dn=false;

anzeige();
  }

}
```